EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

SMART CURTAIN

Ali Hikmet Keklik

180151

Smart curtain, as the name suggests, is a curtain that can be opened automatically without any effort, can be used in home, office etc. environments and makes life easier. A smart curtain that is sensitive to sunlight and can be controlled remotely.

Supervisor

Assoc.Prof.Dr. Cem Burak Kalyoncu

09/06/2023

Table Of Contents

1.Introduction

Smart curtains are a type of window treatment that can be opened and closed automatically, usually using motors or other automation systems. The smart curtain system in this project can be used in a variety of environments, including homes and offices, and allows users to remotely control the curtain or use sensors that detect sunlight or other stimuli. Designed to make life easier by giving The smart curtain can also be programmed to open and close at specific times or in response to other triggers such as temperature or sunlight.

1.1 Problem definition

The smart curtain system, which can be remotely controlled via the website, allows users to open and close their curtain with a single button without standing up and using power. In addition, the system should be able to detect the presence of sunlight and automatically adjust the position of the blinds accordingly, helping people wake up easily, helping to regulate the temperature in a room with the sun's rays.

Example-Problems :

- Being lazy to stand up to open and close the curtain.
- Waking up in a dark environment.
- The curtain are closed when the sun is shining outside and the room gets cold.

4.1 Goals

- Develop a user-friendly web-page that allows users to easily open and close their curtain with the touch of a button.
- Design a system that is reliable and able to function consistently in a variety of settings, including homes and offices.
- Integrate the system with sensors that can detect the presence of sunlight and adjust the position of the curtain accordingly.
- Integrating temperature sensors inside the room and outside the room. So If the air inside is cold and outside is warm, the curtain will open automatically. When it's the other way around, the curtain will close.
- Algorithm that automatically turns on and off at the specified time and present it to the user in the web interface.

- Program the system to open and close the curtain at specific times or in response to other stimuli, such as temperature.
- Test the system thoroughly to ensure that it meets all of the specified requirements and functions as intended.

2. Literature Survey

It is difficult to determine the exact date of the first smart curtain, as there have likely been many different prototypes and innovations in this field over the years. However, smart curtain have become more widely available and popular in recent years with the proliferation of smart home technology. Some early examples of smart curtain or window treatments that can be controlled remotely include the Somfy myLink, which was released in 2013, and the Lutron Serena, which was released in 2014. Before comparing with other smart curtain let's go over our own smart curtain. The smart curtain system that I am going to develop can be controlled through a web interface and integrated with sensors that can detect the presence of sunlight, temperature. This system is intended to be user-friendly and reliable, and it should be able to function consistently in a variety of settings.

Compare1 In 2014, Serena Smart Curtain were released. One aspect of these curtain that is similar to our Smart Curtain is that the components that make the curtain smart are integrated into the curtain itself, rather than being an add-on piece. Another similarity is the ability to control the curtain remotely. However, our Smart Curtain system has the additional feature of being able to be controlled through a web interface using any device with internet access. In contrast, Serena Smart Curtain are a more basic smart curtain system, only offering remote control functionality. Our system has a range of additional features, such as the ability to open and close the curtain based on sunlight or temperature levels, as well as the option to set specific times for the curtain to open and close. [1]

Compare2: Somfy mylink: Somfy's smart curtain system is different from our Smart Curtain in that it is an add-on piece that can be hung on an existing curtain to make it smart. Like our system, Somfy's system can be controlled remotely using a smartphone or tablet but it has application. However, our system has a web interface using any device with internet access. Our system offer the ability to set specific times

for the curtain to open and close. Our system 3 also has additional features such as the ability to open and close the curtain based on sunlight or temperature levels. [2]

Compare3 : SwitchBot's smart curtain system. This is the smart curtain that is the most similar to our smart curtain but it still has a lot of differences. It is similar to our Smart Curtain in that it can be controlled remotely using a smartphone or tablet. The difference is it has mobile application and we have website. Like our system, it also offers the ability to set specific times for the curtains to open and close. However, SwitchBot's system is only compatible with side-opening curtain, while our system can be used roller curtain. In addition, SwitchBot's system is an add-on piece that can be attached to the curtain, similar to Somfy's system, while our system has the components integrated into the curtain itself. Both SwitchBot's and our system allow for remote control. Our system also has additional features such as the ability to open and close the curtain based on sunlight or temperature. SwitchBot's system is also compatible with Alexa, which allows users to control it using voice commands. [3]

Compare4 : SONTE Film is a technology that allows the color of the film to change from transparent to non-transparent. I'm adding this to the comparison because it's also a kind of smart window covering system. The similar thing can be controlled remotely from a smart device. [4]

3. Background Information

3.1 Required & Used software

- **HTML :**
  Hyper Text Markup Language is a standard markup language used for creating web pages.
- **CSS :**
  Cascading Style Sheets is a style sheet language used for describing the look and formatting of a document written in HTML.
- **Bootstrap:**
  Bootstrap is a free front-end framework for faster and easier web development. It includes HTML and CSS based design templates for typography, forms, buttons, tables, etc.
- **C++ :**
  C ++ is known for its ability to handle complex tasks and perform well in resource-constrained environments. It is often used for applications that require high levels of performance or that need to interact with low-level hardware components.
- **Arduino :**
  The Arduino software is a free, open-source platform that includes a development environment and libraries for creating software that can interact with hardware devices.

3.2 Other software

- **Bitbucket :**

  Bitbucket is a web-based version control repository hosting service for source code and development projects.

- **Git :**
  Bitbucket is a web-based version control repository hosting service for source code and development projects.
- **Visual Studio Code :**

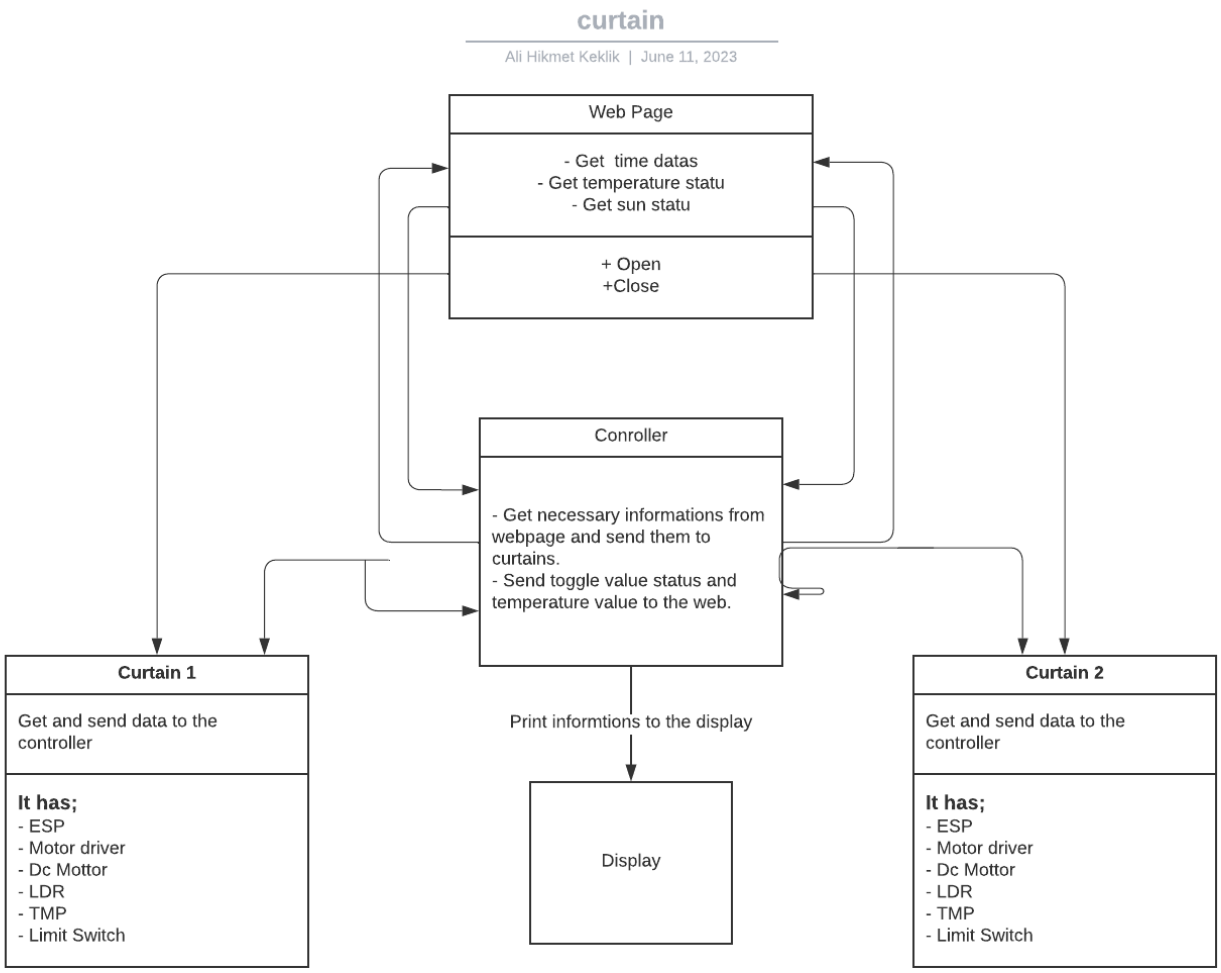
Visual Studio Code is a code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, source control, and code refactoring.

3.3 Hardware

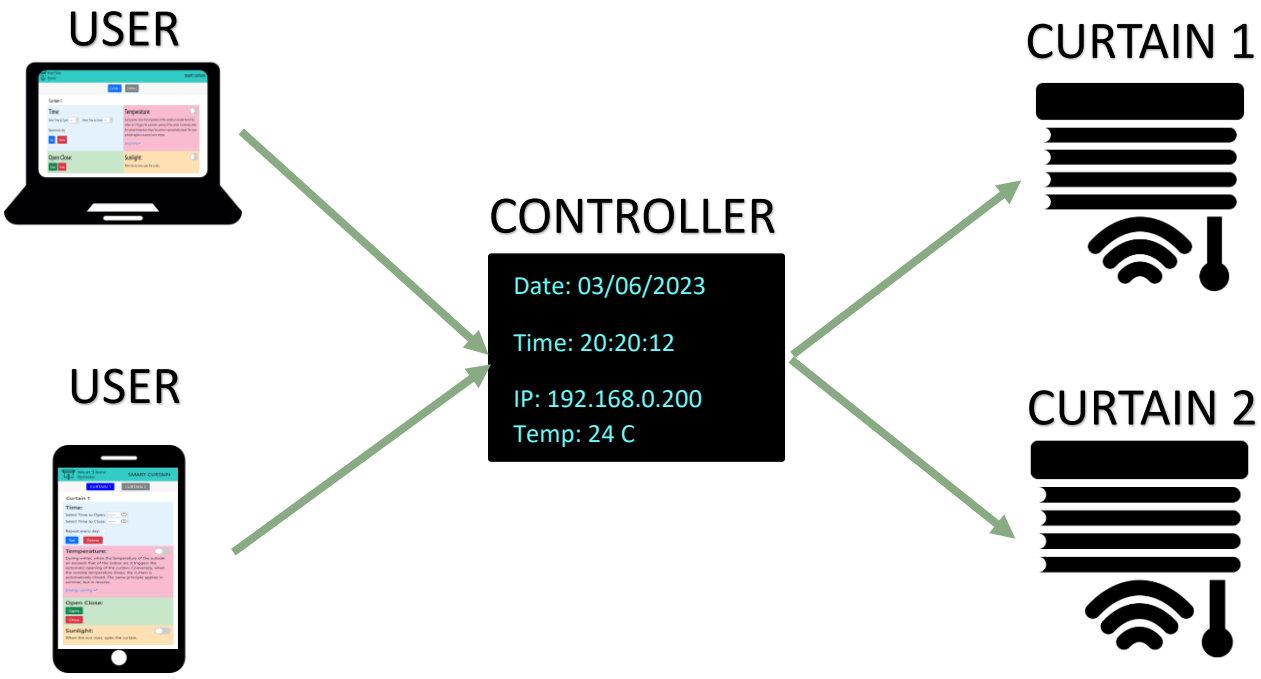
- **NodeMCU V3 ESP8266 ESP :**
  The NodeMCU V3 ESP8266 ESP is a low-cost microcontroller with built-in WiFi support. It can be used to control devices wirelessly over the internet.
- **12V 12mm 120 RPM DC Motor :**
  A DC motor is an electric motor that runs on direct current (DC) electricity. The 12V 12mm 120 RPM version is a small, low-voltage motor that rotates at a speed of 120 revolutions per minute.
- **Mikro Limit Switch :**
  A limit switch is a mechanical switch that is activated by the motion of an object. The Mikro limit switch is a small, high-precision switch that can be used to detect the end positions of moving objects.
- **L298N Motor Driver :**
  The L298N Motor Driver is a dual H-bridge motor driver that can be used to control the speed and direction of two DC motors.
- **LM335AZ TMP Sensor :**
  A temperature sensor that can be used to measure the temperature of a device or environment.
- **LDR :**
  A light-dependent resistor, also known as a photoresistor, is a type of resistor that changes resistance based on the amount of light it is exposed to. It can be used to detect the presence of sunlight or other sources of light.

4. Design Documents

4.1 Data flow diagram



4.2 Context Diagram

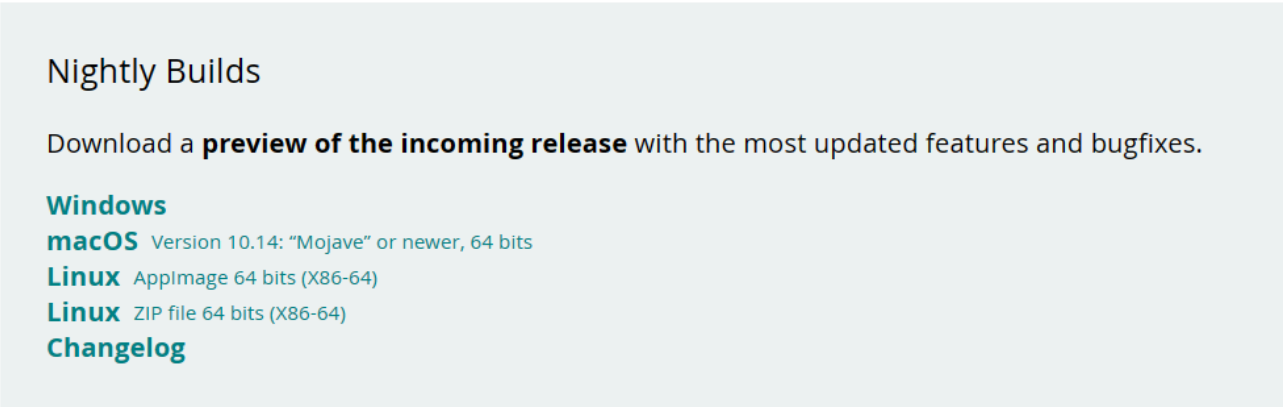


5. Methodology

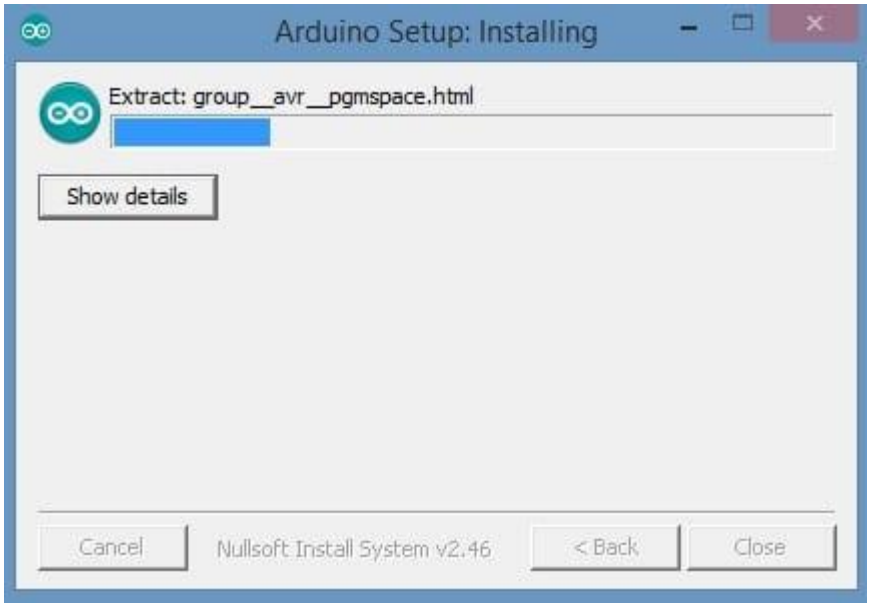
5.1 Installing Arduino

➔

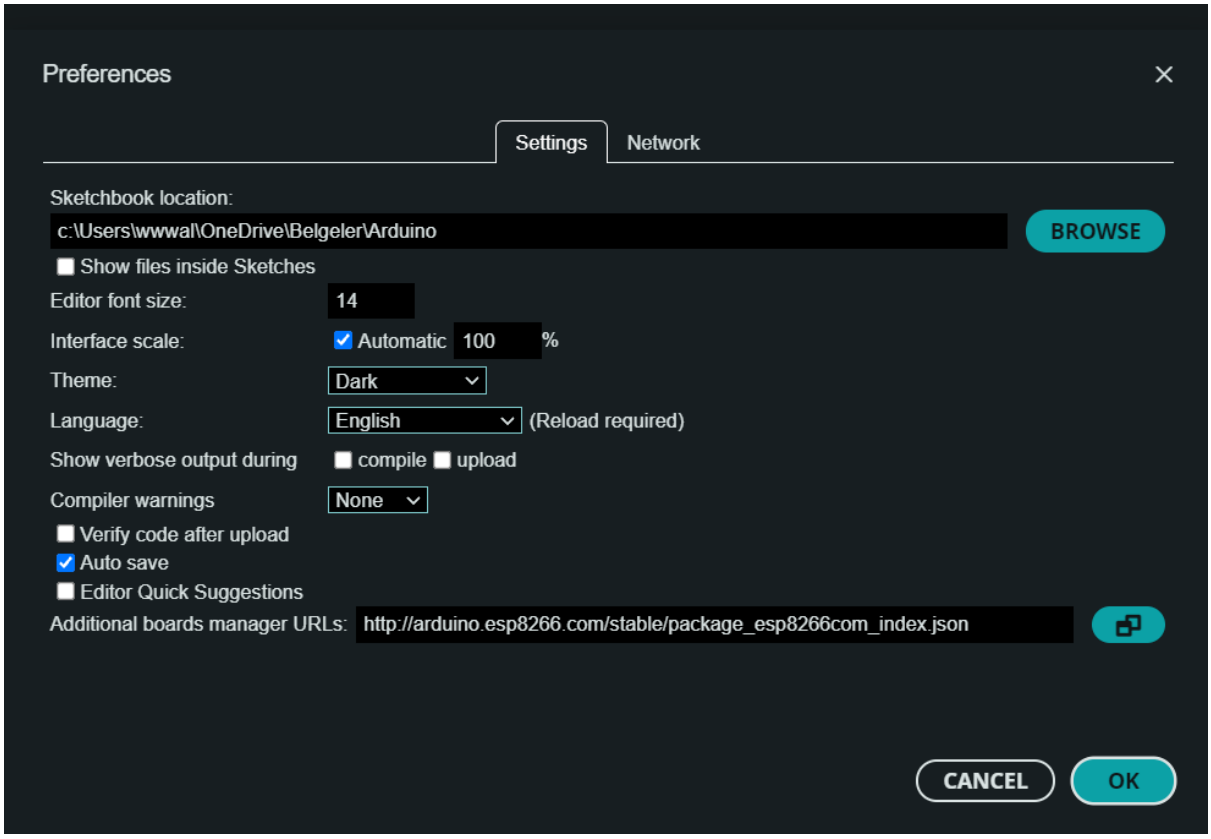
➔ With following website [Software | Arduino](), We click on the appropriate option according to the operating system and download the exe file.



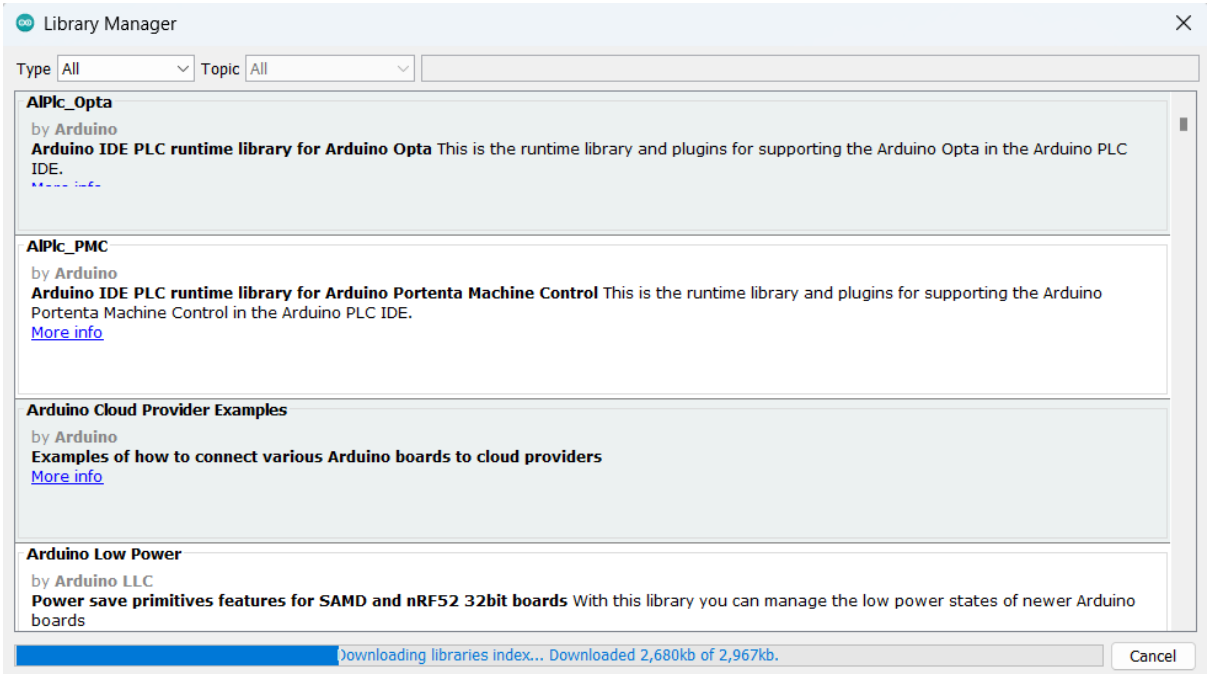
➔ We open the exe file and do the installation.



➔ In order to use esp8266, we enter the necessary url in the preferences section.



➔ Downloading libraries that we will use

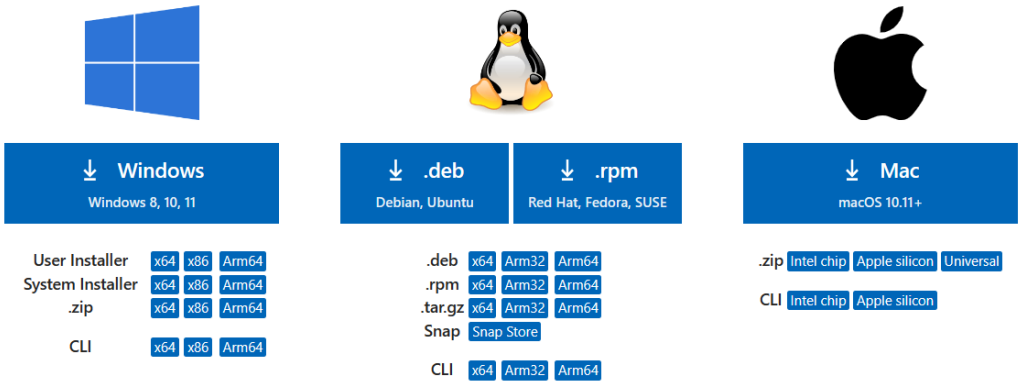


In our case we used flowing libraries.

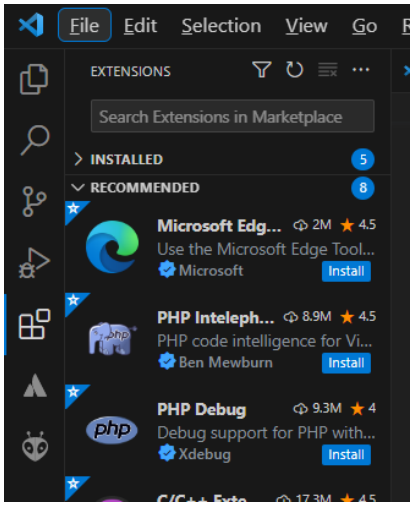
```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <ArduinoJson.h>
#include <WiFiClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <TimeLib.h>
#include <ezButton.h>
#include <EEPROM.h>
```

5.2 Installing VS Code

➔ Downloading VS code with the following url. [Download Visual Studio Code - Mac, Linux, Windows]



➔ We do the installation and download the necessary extensions.



5.3 WebPage

The web page is designed to allow the user to interact with the curtain from any device and to change the settings related to a particular feature. The initial component of the page consists of a navigation bar containing specific elements like a logo and a prominent banner. The layout of the page is built using a CSS framework known as Bootstrap.

In a particular section there is a div with buttons that give the option to switch between two different curtains. When a button is selected, the corresponding screen becomes visible. We do these operations with javascript.

Another element present on the page is a designated area where users can configure settings related to the time setting. In this area, the user can set the opening and closing times and repeat them every day. WE used input type time for this and we used checkbox to ask if it will work every day. When the set button is pressed, the specified time settings are sent to the controller and when the delete button is pressed, the setting will be deleted.

The second part is the area where the temperature status is controlled. Inside this area is a toggle button that shows and changes the temperature status. It also has a text showing the temperature value. In this case, the toggle button is used to send the temperature status to the controller.

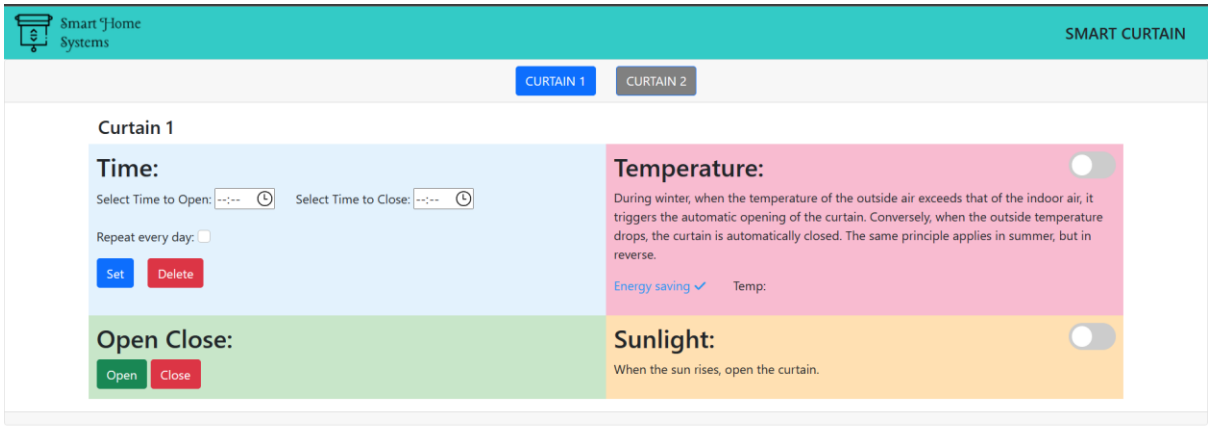
The Open and Close buttons are also located on the page to perform the opening and closing operations of the curtains. When these buttons are clicked, commands to the corresponding curtain are sent directly to the curtain.

Another div where the sun status is checked is also on the page. In this section there is a toggle button, as in the case of temperature. When this button is clicked, the sun status is sent to the controller.

Lastly, the page incorporates a function that enables smooth transitions between curtain settings. Additionally, there is a feature that automatically disables the toggle buttons for sunlight and temperature when specific time settings are entered.

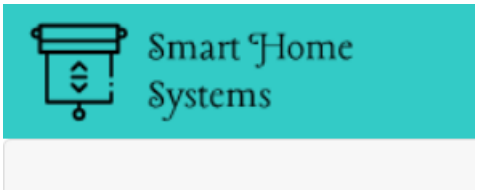
In conclusion, this web page provides users with the ability to customize curtain settings, control the curtains themselves, monitor temperature and sunlight conditions, as well as modify the associated configurations.

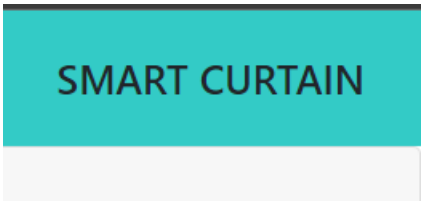
5.3.1 Webpage Picture



5.3.2 Creating navbar with logo and title

```
<nav style="background-color: rgb(51, 203, 198);" class="navbar bg-body-
tertiary">
        <div class="container-fluid ">
            <img src="https://i.ibb.co/BKstZzX/logo.png" alt="Smart Home
Systems">
            <span class="navbar-brand mb-0 h1">SMART CURTAIN</span>
        </div>
    </nav>
```





5.3.3 Buttons and alerts

➔ Buttons that we can switch between screen one and two and alerts that will be shown when some actions are performed.

```
<div style="text-align: center;" class="card-header">
                <button id="firstCurtain" name="curtain1" class="btn btn-
primary c1"
                    onclick="showCurtain('curtain1')">CURTAIN 1</button>
                    
                <button style="background-color: gray;" id="secondCurtain"
name="curtain2" class="btn btn-primary c2"
                    onclick="showCurtain('curtain2')">CURTAIN 2</button>
            </div>
```

```
            <div id="alertDiv1" class="alert alert-success alert-dismissable"
style="display: none;">
                <button id="closeBtn" type="button" class="close" aria-
hidden="true"
                    onclick="hideAlert1()">&times;</button>
                Success! Time is set.
            </div>
            <div id="alertDiv2" class="alert alert-warning alert-dismissable"
style="display: none;">
                <button id="closeBtn" type="button" class="close" aria-
hidden="true"
                    onclick="hideAlert2()">&times;</button>
                Set time deleted!
            </div>
```

5.3.4 Getting time data

➔ Using input type time to get time information, checkbox for repeat everyday and buttons that will set and delete.

```
<form action="" method="post">
     <div class="mb-3">
     <label for="timer1open" class="form-label">Select Time to Open:</label>
     <input type="time" id="timer1open" name="timer1open" class="custom-time-
input">
         
     <label for="timer1close" class="margin-left-5 form-label">Select Time to
          Close:</label>
     <input type="time" id="timer1close" name="timer1close" class="custom-
time-input">
     </div>
     <div class="mb-3">
     <label for="repeatDaily" class="form-check-label">Repeat every
day:</label>
     <input type="checkbox" id="repeatDaily" name="repeatDaily" class="form-
check-input">
     </div>
     <button type="button" id="saveTimeBtn1" class="btn btn-primary"
     onclick="executeTime1Functions()">Set</button>
       
     <button type="button" class="btn btn-danger" id="deleteTimeBtn1"
name="deleteTimeBtn1"
     onclick="executeTime1DeleteFunctions()">Delete</button>
</form>
```

5.3.5 Sending Time Data to Controller

➔ Sending datas using Ajax

```
function executeTime1Functions() {
            var timer1close = document.getElementById("timer1close").value;
            var timer1open = document.getElementById("timer1open").value;
            var repeatDaily = document.getElementById("repeatDaily").checked;
            var xhttp = new XMLHttpRequest();
            xhttp.open("GET", "/set-timer1?timer1open=" + timer1open +
"&timer1close=" + timer1close + "&repeatDaily=" + repeatDaily, true);
            xhttp.send();
            showAlert1();
        }
        function executeTime1DeleteFunctions() {
            var xhttp = new XMLHttpRequest();
            xhttp.open("GET", "/set-delete1", true);
            xhttp.send();
            showAlert2();
        }
```

5.3.6 Getting Temperature Status

➔ Get status of temperature using checkbox. With CSS it will be toggle button.

```
<div id="temperature1" class="col-md-6 custom-col" style="background-color:
#f8bbd0;">
    <h2>Temperature:
        <label class="float-end switch">
            <input type="checkbox" id="tempToggleButtonCurtain1"
name="tempToggleButtonCurtain1"
                onclick="sendTempToggleValue()">
            <span class="slider round"></span>
        </label>
    </h2>
    <p>During winter, when the temperature of the outside air exceeds that of
the indoor air,
        it triggers the automatic opening of the curtain. Conversely,
        when the outside temperature drops, the curtain is automatically
closed.
        The same principle applies in summer, but in reverse.
    </p>
    <p style="color: #2196F3; display: inline-block;">Energy saving <i
        class="fa-solid fa-check"></i></p>
<p style="display: inline-block; margin-left: 30px;">Temp: <span style="color:
black"
        id="temperatureValue1"></span></p>
```

```
</div>
```

5.3.7 Sending temp status to controller

➔ Sending the status of temperature to the controller using ajax

```
function sendTempToggleValue2() {
            var toggleValue =
document.getElementById("tempToggleButtonCurtain2").checked;
            var xhttp = new XMLHttpRequest();
            xhttp.open("GET", "/handleTempToggleValueCurtain2?value=" +
toggleValue, true);
            xhttp.send();
        }
```

5.3.8 Open and Close Buttons

➔ Open and close buttons. Using bootstraps success and danger class.

```
<div class="d-grid gap-2 d-md-block">
    <button  id="open2" name="opne1" class="btn btn-success" type="button"
onclick="openCurtain2()">Open</button>
    <button  id="close2" name="close1" class="btn btn-danger" type="button"
onclick="closeCurtain2()">Close</button>
</div>
```

5.3.9 Send open/close commands

➔ Send open and close command directly according to the ip addresses of the curtains.

```
function openCurtain2() {
            var xhttp = new XMLHttpRequest();
            xhttp.open("GET", "http://192.168.0.115/open-curtain2", true);
            xhttp.send();
        }
        function closeCurtain2() {
            var xhttp = new XMLHttpRequest();
            xhttp.open("GET", "http://192.168.0.115/close-curtain2", true);
            xhttp.send();
        }
```

5.3.10 Getting Sun Status

➔ Get status of sun using checkbox. With CSS it will be toggle button.

```
<div id="sun2" class="col-md-6 custom-col" style="background-color: #ffe0b2;">

     <h2>Sunlight:<label class="float-end switch">
     <input type="checkbox" id="sunToggleButtonCurtain2"
name="sunToggleButtonCurtain2"onclick="sendSunToggleValue2()">
     <span class="slider round"></span>
     </label></h2>
     <p>When the sun rises, open the curtain.</p>
</div>
```

5.3.11 Sending sun status to controller

➔ Sending the status of sun to the controller using ajax

```
function sendSunToggleValue() {

            var toggleValue =
document.getElementById("sunToggleButtonCurtain1").checked;
            var xhttp = new XMLHttpRequest();
            xhttp.open("GET", "/handleSunToggleValueCurtain1?value=" +
toggleValue, true);
            xhttp.send();
        }
```

5.3.12 Function to switch between curtains

```
 function showCurtain(curtainId) {
            var curtain1 = document.getElementById('curtain1');
            var curtain2 = document.getElementById('curtain2');

            if (curtainId === 'curtain1') {
                curtain1.removeAttribute('hidden');
                curtain2.setAttribute('hidden', 'true');
                secondCurtain.style.backgroundColor = 'gray';
                firstCurtain.style.backgroundColor = 'blue';
            } else if (curtainId === 'curtain2') {
                curtain1.setAttribute('hidden', 'true');
                curtain2.removeAttribute('hidden');
                firstCurtain.style.backgroundColor = 'gray';
                secondCurtain.style.backgroundColor = 'blue';
            }
```

```
        }
```

5.3.13 Close Sun Toggle

➔ Function that turns off the sun button when time is set.

```
//  Curtain 1 Function to turn off sun toggles when the save button is clicked
        function turnOffToggles1() {
            var temperature1Switch = document.querySelector('#temperature1
input[type="checkbox"]');
            var sun1Switch = document.querySelector('#sun1
input[type="checkbox"]');

            sun1Switch.checked = false;
        }

        // Add event listener to the save button
        var saveTimeBtn1 = document.querySelector('#saveTimeBtn1');
        saveTimeBtn1.addEventListener('click', turnOffToggles1);
```

5.4 Arduino

It is a system that provides communication between a control device and curtains. The controller receives the data entered by the user from the web page and directs it to the curtains. It also receives data from sensors to detect environmental conditions such as temperature and sun setting.

Controller uses a struct to store curtain data. This structure contains information such as curtain opening and closing times, repeating options, temperature and sun settings. This data is saved in EEPROM and read from EEPROM when necessary.

The project connects the controller with the modem using a WiFi connection. The user must connect to this network to check the curtain settings. The controller makes the connection in the setup function.

An HTML file is read so that the web page can run on the local server. This file provides the user interface and allows the user to adjust the curtain settings.

When the user clicks the timer button, the controller receives this data and saves it to the EEPROM. Likewise, if the temperature and sun setting buttons are clicked, the relevant data is received and the states are changed automatically.

Controller sends curtain data to curtains using JSON document. The necessary data is written and serialized in this document. The pitch receives this data from the controller, deserializes it and adjusts the pitch state accordingly.

The controller has an LCD display. This screen shows information such as date, time, IP address and temperature. Also, the controller has a temperature sensor outside the curtain. This sensor measures the outside temperature and is received by the controller.

Curtains receive data from the controller inside the main loop. This data comes in JSON format and is deserialized by the curtain. In this way, information such as curtain opening and closing times, repeat option, temperature and sun settings become available to the curtain.

Curtains are controlled by the user via the web page. The curtain moves when the on and off buttons are pressed. Also, the curtain has temperature sensors for indoor and outdoor and a light sensor for outdoor. An analog multiplexer is used to read these sensors.

The motor is controlled as the curtain opens and closes. The curtain opens or closes according to the movement direction of the motor. The opening speed is one click faster than the closing speed because the effect of gravity is taken into account.

There are limit switches on the curtains. When the limit switch is pressed, the motor is stopped and the curtain state is saved in the EEPROM. Thus, the last state of the curtain can be remembered even after the power is turned off.

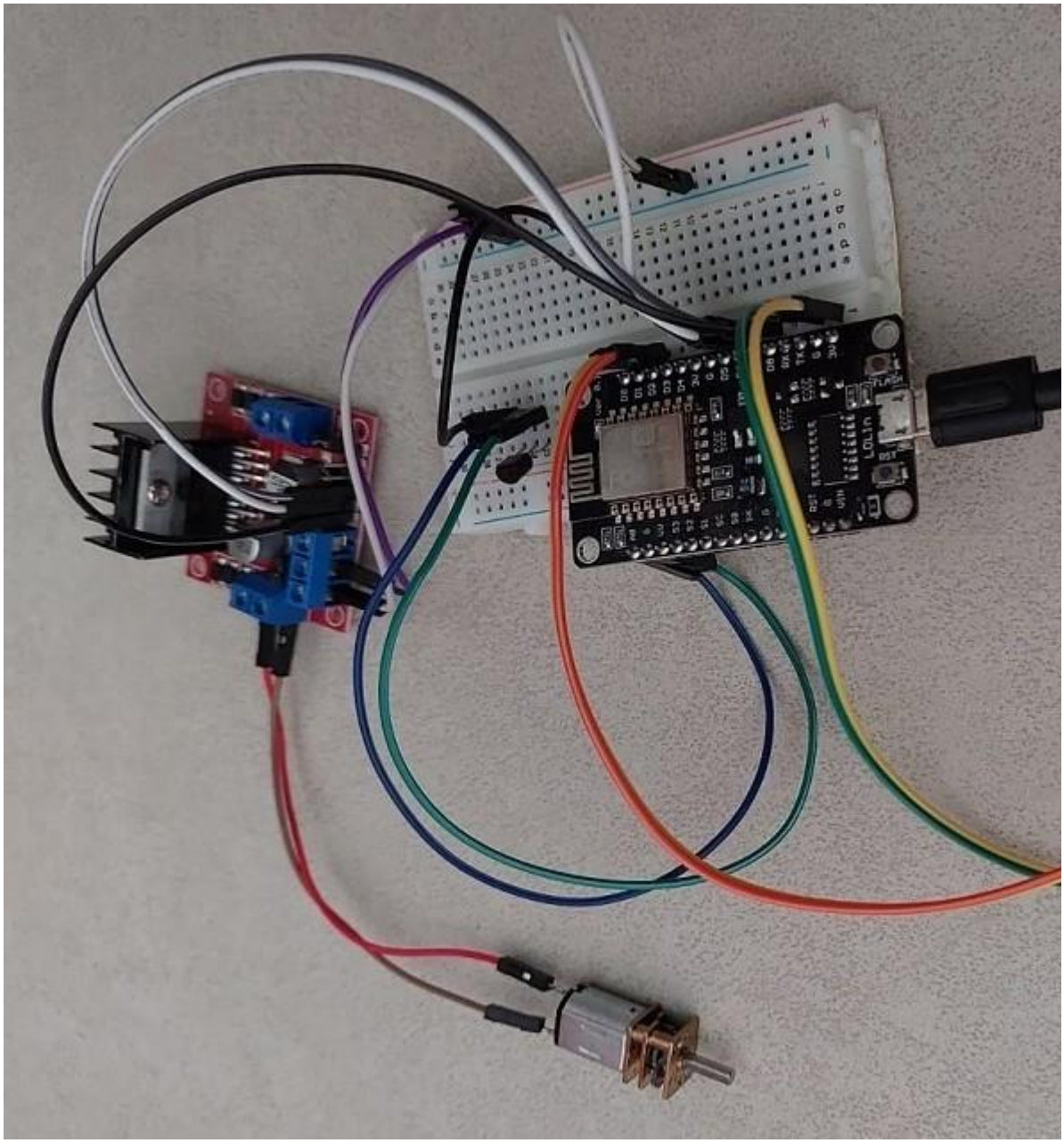
In the system that works in this way, the controller receives the user's data, transmits them to the curtains, controls their movements and detects the environmental conditions. Curtains, on the other hand, act according to the data they receive from the controller and measure the environmental conditions through sensors.

5.4.1 Connections

DC Motor – Motor Driver - ESP

In our project, we have successfully established the necessary connections between the ESP , motor driver, and DC motor. To control the motor, we utilized specific pins on the ESP and made appropriate connections to the motor driver. The enA pin on the motor driver is connected to digital pin D8 on the ESP microcontroller, which allows us to control the

motor's speed using PWM signals. Additionally, we connected the in1 and in2 pins of the motor driver to digital pins D5 and D6, respectively, on the ESP . These pins control the direction of the motor rotation. To ensure reliable connections, we used jumper cables to establish the electrical connections between the microcontroller and the motor driver, and we employed a breadboard to conveniently organize and secure the connections. With this setup, we were able to successfully drive the DC motor, enabling precise control over its speed and direction of rotation.



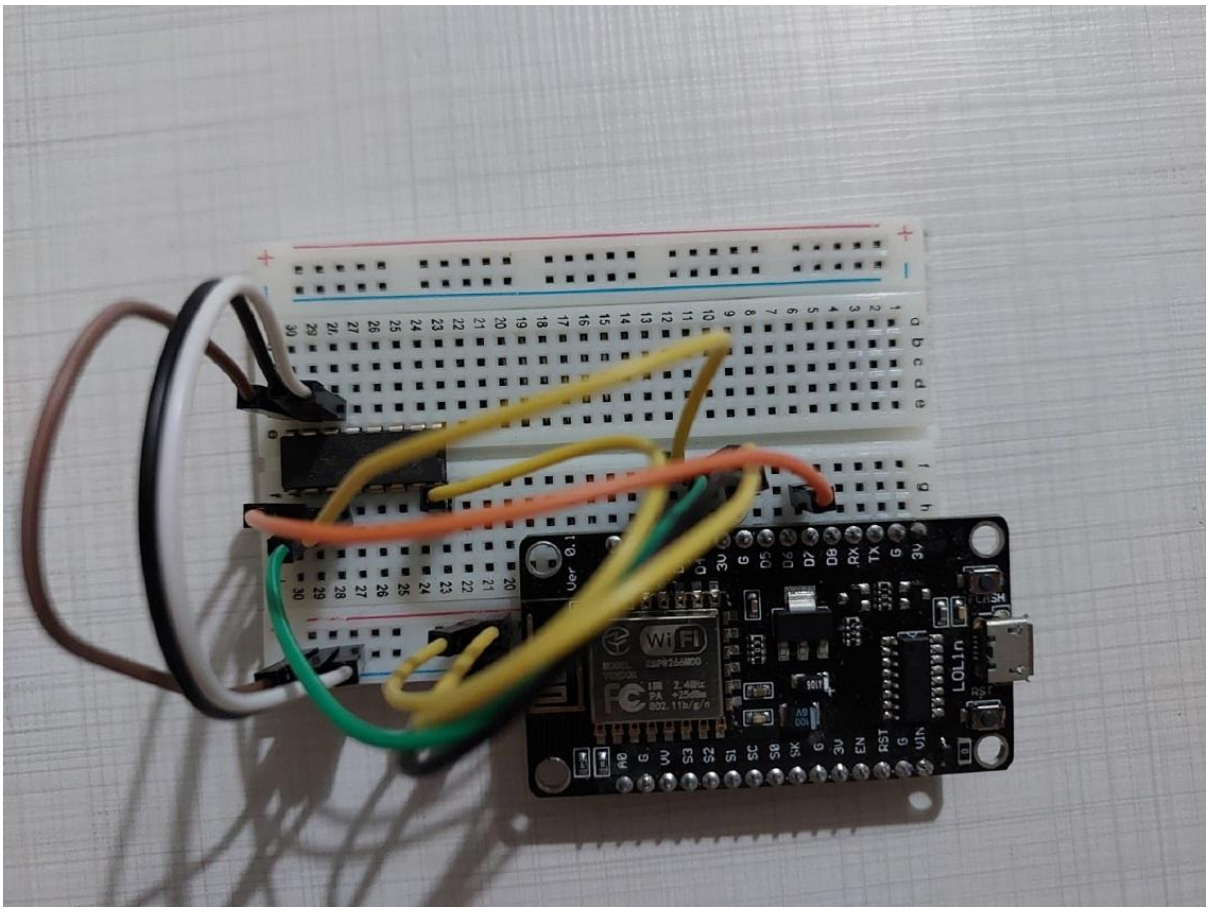
ESP8266 – 4051BP

In our project, we have established connections between an analog multiplexer (4051BP), and the ESP8266. The analog multiplexer is controlled by three pins on the ESP , namely controlPinA (connected to D3), controlPinB (connected to D4), and controlPinC (connected

to D7). These pins allow us to select the appropriate sensor channel for reading analog input values. The common output of the 4051BP is connected to the analog input pin A0 on the ESP microcontroller. During the setup phase, we set the control pins as outputs to control the multiplexer.

To read sensor values, we implemented the updateSensorValues() function. Within this function, a loop is used to iterate through the three sensor channels. The selectSensorChannel() function is called to set the appropriate control pin states for the selected channel. For channel 0, the analog input value from the inside temperature sensor is read using analogRead(), and the value is then converted to temperature in degrees Celsius. Similarly, for channel 1, the analog input value from the outside temperature sensor is read and converted to temperature. For channel 2, the input from an LDR (Light Dependent Resistor) sensor is read and stored. These readings are obtained by multiplexing the common output of the 4051BP.

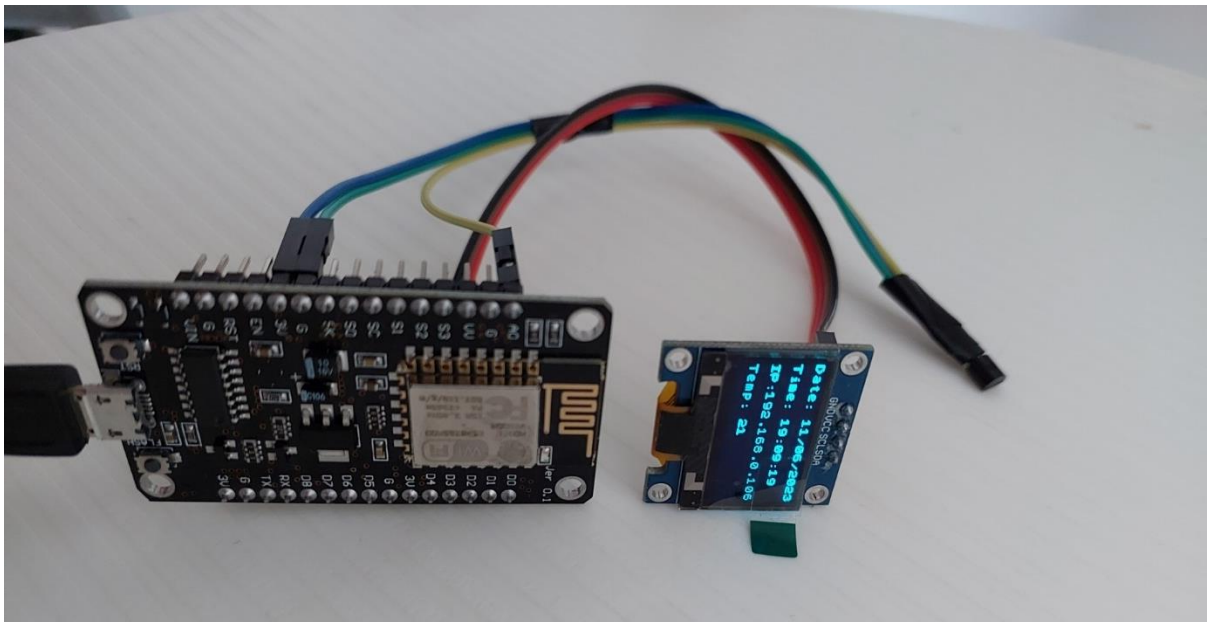
By utilizing this setup, we are able to selectively read values from multiple sensors using a single analog input pin on the ESP microcontroller. This approach offers efficient sensor data acquisition and simplifies the overall circuit design.



ESP-LCD-LM335AZ

We have an ESP8266-OLED display module and an LM335az temperature sensor. The connections for the OLED display are standard and do not require any specific instructions(V to V, G to G, SCL to SCL, SDA to SDA). As for the LM335az temperature sensor, it is connected as follows: the GND (Ground) pin of the temperature sensor is connected to the GND (Ground) pin on the ESP8266. Vin pin of the temperature sensor is connected to the 3.3V power supply on the ESP8266, and the remaining pin of the temperature sensor is connected to the analog input pin A0 on the ESP8266.

After we did the connections, we can obtain temperature readings from the LM335az sensor and display them on the OLED module using the ESP8266 microcontroller. The LM335az temperature sensor provides analog output that can be read by the ESP8266's analog-to-digital converter through the A0 pin. We convert the analog input value to the celcius.The OLED display, on the other hand, allows us to see the temperature readings in a user-friendly screen.



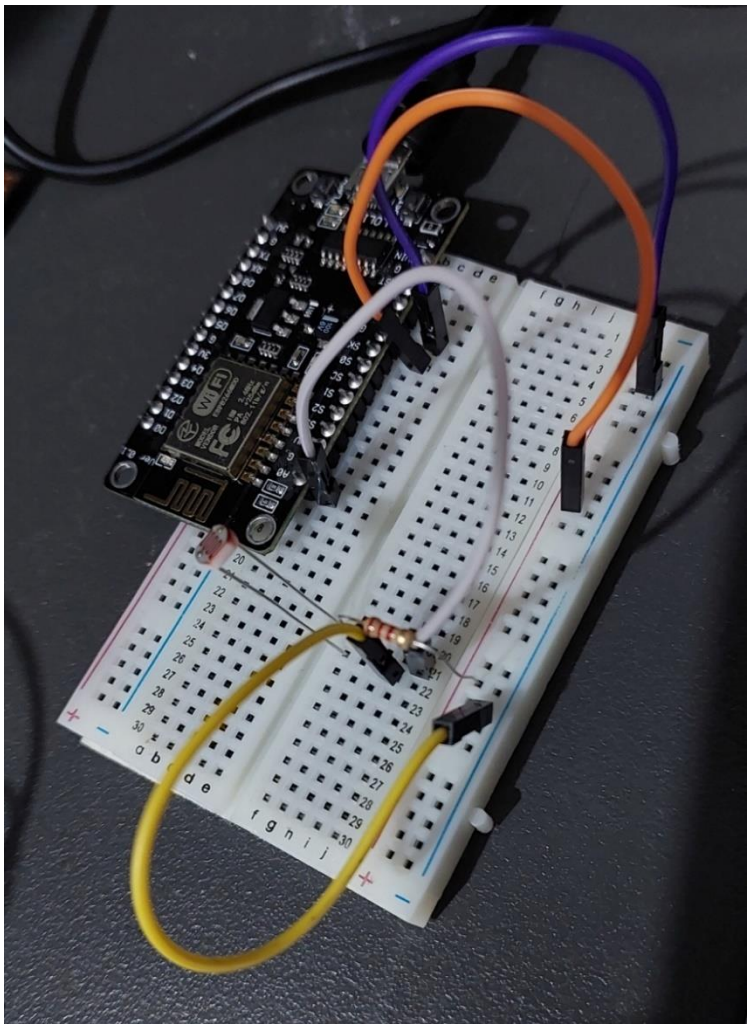
ESP - LDR

In our project, we have integrated an LDR (Light-Dependent Resistor) with the ESP8266 microcontroller using a 4051 analog multiplexer. The LDR is a type of resistor whose resistance varies according to the intensity of light falling on it. This allows us to measure and monitor the ambient light levels in our project.

To establish the connection, we have connected the LDR to channel 2 of the 4051 analog multiplexer. The LDR is connected using a 2.2k resistor, which forms a voltage divider circuit. The GND (Ground) pin of the LDR is connected to the GND (Ground) pin on the ESP8266, other pin of the LDR is connected to the analog input pin A0 on the ESP8266. The junction between the LDR and the 2.2k resistor is connected to the common output pin of the 4051 multiplexer.

By employing this setup, we can read the analog voltage output from the LDR through the A0 pin of the ESP8266. The 4051 analog multiplexer allows us to select the desired channel. In this case channel 2 which is connected to the LDR. This multiplexing capability enables us to switch between 8 different sensors connected to the multiplexer and retrieve the corresponding analog data.

With this connection, we can effectively capture the light intensity detected by the LDR and utilize it in our project. The ESP8266 microcontroller can process this analog input and perform various actions or trigger events based on the measured light levels. This functionality enables us to incorporate light-sensing capabilities into our project, making it more versatile and responsive to the surrounding environment.



5.4.2 Controller

5.4.2.1 Struct Type

➔ Creating struct that will store curtain datas.

```
  int storedHourOpen;
  int storedMinuteOpen;
  int storedHourClose;
  int storedMinuteClose;
  bool repeatEveryDay = false;

  // toogle values we get from user (webpage)
  bool tempToggleValueCurtain = false;
  bool sunToggleValueCurtain = false;
  bool timeSet = false;
  int storedDay;
  int storedMonth;
  int storedYear;
} curtain;
```

5.4.2.2 Eeprom

First initialize EEPROM with the required size “EEPROM.begin(50); “

➔ Committing informations to the eeprom.

```
EEPROM.put(eepromAddress, timerOpenHourMinute2);
eepromAddress += sizeof(timerOpenHourMinute2);
EEPROM.put(eepromAddress, timerCloseHourMinute2);
eepromAddress += sizeof(timerCloseHourMinute2);
EEPROM.put(eepromAddress, curtain2.repeatEveryDay);
eepromAddress += sizeof(curtain2.repeatEveryDay);
EEPROM.put(eepromAddress, curtain2.timeSet);
eepromAddress += sizeof(curtain1.timeSet);
EEPROM.put(eepromAddress, storedDay);
eepromAddress += sizeof(storedDay);
EEPROM.put(eepromAddress, storedMonth);
eepromAddress += sizeof(storedMonth);
EEPROM.put(eepromAddress, storedYear);
eepromAddress += sizeof(storedYear);
EEPROM.commit();
```

➔ Getting informations from eeprom when we need it

```
EEPROM.get(32, curtain2.repeatEveryDay);
EEPROM.get(33, curtain2.timeSet);
EEPROM.get(34, curtain2.storedDay);
EEPROM.get(38, curtain2.storedMonth);
EEPROM.get(42, curtain2.storedYear);
EEPROM.get(46, curtain2.tempToggleValueCurtain);
EEPROM.get(47, curtain2.sunToggleValueCurtain);
```

5.4.2.3 Connecting Wifi

➔ We define ssid and the password that of the modem that we connected. It works only with this modem. The person who will control the curtain must be connected to this network

```
const char* ssid = "ALFA";
const char* password = "167349abc";
```

➔ Then we connect in the setup function

```
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
}

Serial.println("Connected to WiFi");
Serial.print("Local IP: ");
Serial.println(WiFi.localIP());
```

5.4.2.4 Read File

➔ Webpage should be work on the localhost so we are handling this with reading a html file.

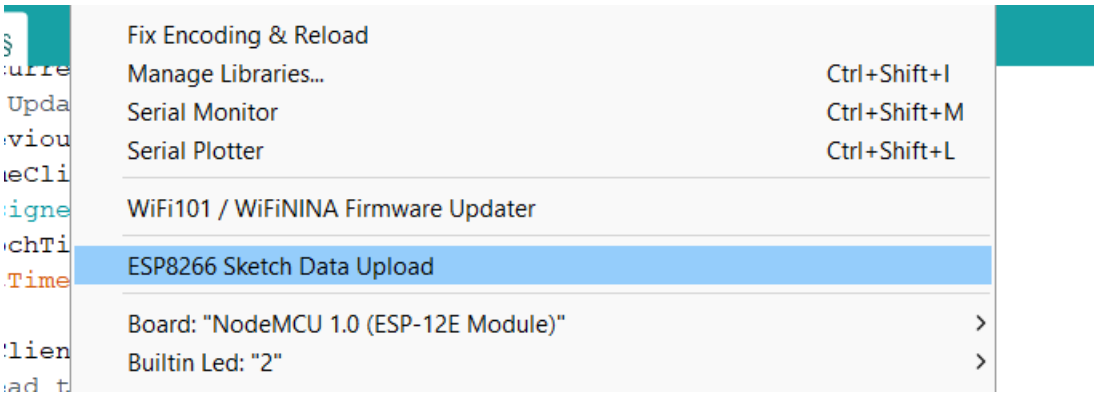
```
//read the html file
void handleRoot() {
  String htmlPage = readFile("/index.txt");
  server.send(200, "text/html", htmlPage);
}


String readFile(const char* path) {
  File file = SPIFFS.open(path, "r");
  if (!file) {
    Serial.println("Failed to open file for reading");
    return String();
  }

  String content;
  while (file.available()) {
    content += (char)file.read();
  }

  file.close();
  return content;
}
```

The following part is uploading file to the Arduino IDE.



5.4.2.5 Get Timer

➔ When user click set time button and if user enters two time in the input field following function will be called.

```
void handleTimer1() {

  if (server.hasArg("timer1open") || server.hasArg("timer1close")) {
    eepromAddress = 0;
    timeClient.update();
    server.handleClient();

    unsigned long epochTime = timeClient.getEpochTime();
    epochTime += 3 * 3600;
    setTime(epochTime);
    curtain1.timeSet = true;
    curtain1.sunToggleValueCurtain = false;

    String repeat = server.arg("repeatDaily");
    if (repeat == "true") {
      curtain1.repeatEveryDay = true;
    } else {
      curtain1.repeatEveryDay = false;
    }

    String timerOpenValue = server.arg("timer1open");
    String timerCloseValue = server.arg("timer1close");
    Serial.println("Timer open set to: " + timerOpenValue);
    Serial.println("Timer close set to: " + timerCloseValue);

    int timerOpenHour = timerOpenValue.substring(0, 2).toInt();
    int timerOpenMinute = timerOpenValue.substring(3, 5).toInt();
    int timerCloseHour = timerCloseValue.substring(0, 2).toInt();
```

➔ After we get the datas we store them in eeprom in the same function.

➔

5.4.2.6 Get Temperature

➔ When user click the temperature toggle button on the webpage. This function will be called and the status will change automatically.

```
void handleTempToggleValueCurtain1() {
  eepromAddress = 22;
  oled.clearDisplay();
  if (server.hasArg("value")) {
    String value = server.arg("value");
    Serial.println("Received value: " + value);
    if (value == "true") {
      curtain1.tempToggleValueCurtain = true;
    } else {
      curtain1.tempToggleValueCurtain = false;
    }
    EEPROM.put(eepromAddress, curtain1.tempToggleValueCurtain);
    EEPROM.commit();
    server.send(200, "text/html", "Temp toggle value set");
  }
}
```

5.4.2.7 Get Sun

➔ When user click the sun toggle button on the webpage. This function will be called and the status will change automatically.

```
void handleSunToggleValueCurtain1() {
  curtain1.timeSet = false;
  curtain1.sunToggleValueCurtain = true;
  eepromAddress = 23;
  //oled.clearDisplay();
  if (server.hasArg("value")) {
    String value = server.arg("value");
    Serial.println("Received value: " + value);
    if (value == "true") {
      curtain1.sunToggleValueCurtain = true;
    } else {
      curtain1.sunToggleValueCurtain = false;
    }
    EEPROM.put(eepromAddress, curtain1.sunToggleValueCurtain);
    EEPROM.commit();
    server.send(200, "text/html", "Sun toggle value set");
  }
}
```

5.4.2.8 Send Data to Curtain

➔ Controller should send the curtain datas to the curtains and we handle this with Json Document. First we write every data that we will send then serialize it. The function will be called when curtain ask it.

```
void handleData1() {

  StaticJsonDocument<512> jsonDoc;

  jsonDoc["storedHourOpen"] = curtain1.storedHourOpen;
  jsonDoc["storedMinuteOpen"] = curtain1.storedMinuteOpen;
  jsonDoc["storedHourClose"] = curtain1.storedHourClose;
  jsonDoc["storedMinuteClose"] = curtain1.storedMinuteClose;
  jsonDoc["repeatEveryDay"] = curtain1.repeatEveryDay;
  jsonDoc["tempToggleValueCurtain"] = curtain1.tempToggleValueCurtain;
  jsonDoc["sunToggleValueCurtain"] = curtain1.sunToggleValueCurtain;
  jsonDoc["timeSet"] = curtain1.timeSet;
  jsonDoc["storedDay"] = curtain1.storedDay;
  jsonDoc["storedMonth"] = curtain1.storedMonth;
  jsonDoc["storedYear"] = curtain1.storedYear;

  char jsonData[512];
  serializeJson(jsonDoc, jsonData);

  server.send(201, "application/json", jsonData);
}
```

5.4.2.9 LCD

➔ Controller has a LCD screen and it displays Date, Time, IP Address, Temperature.

```
oled.setTextXY(0, 0);
oled.putString("Date: ");
putNumberWithLeadingZero(day());
oled.putString("/");
putNumberWithLeadingZero(month());
oled.putString("/");
oled.putNumber(year());
oled.setTextXY(2, 0);
oled.putString("Time: ");
putNumberWithLeadingZero(hour());
oled.putString(":");
putNumberWithLeadingZero(minute());
oled.putString(":");
putNumberWithLeadingZero(second());
oled.setTextXY(4, 0);
oled.putString("IP:");
String ip = WiFi.localIP().toString();
oled.putString(ip.c_str());
oled.setTextXY(6, 0);
oled.putString("Temp: ");
putNumberWithLeadingZero(temperature);
```

5.4.2.10 Temperature

➔ Except the curtains, Controller has separate temperature sensor.

```
// Read temperature from LM335AZ sensor
int sensorValue = analogRead(A0);
// Convert analog value to Celsius temperature
temperature = sensorValue * 330 / 1024 - 273;
```

5.4.2.11 Receive Temperature and Status

➔ Controller can get outside temperature and the status of the curtain (open/close).

```
void handleTempAndStatus1() {
  if (server.method() != HTTP_POST) {
    server.send(405, "text/plain", "Method Not Allowed");
    return;
  }

  DynamicJsonDocument json(1024);
  DeserializationError error = deserializeJson(json, server.arg("plain"));

  if (error) {
    server.send(400, "text/plain", "Bad Request");
    return;
  }

  curtain1Temp = json["temperature"];
  curtain1Status = json["curtainStatus"];

  server.send(200, "text/plain", "OK");
}
```

5.4.3 Curtain

5.4.3.1 Get Data From Controller

➔ Inside the main loop we do have function that will call the following function every 10 minutes. So we gat data from controller every 10 seconds. We are not doing this in the main loop because it will get the datas continuously and it will cause some delay problems or data loses

```
void getCurtainDatas() {
  WiFiClient client;
  HTTPClient http;

  // Concatenate the URL
  String url = "http://";
  url += serverIP;
  url += ":";
  url += serverPort;
  url += endpoint;

  http.begin(client, url);
  int httpResponseCode = http.GET();

  if (httpResponseCode == 300) {
    String response = http.getString();
    handleData(response);
  } else {
    Serial.print("Error - HTTP response code: ");
    Serial.println(httpResponseCode);
  }
```

➔ As we mentioned in 5.4.1.8 we were sending datas in Json format. The following function will deserialize it and then will store datas in struct.

```
void handleData(String jsonData) {
  // Parse the JSON data
  DynamicJsonDocument jsonDoc(512);
  deserializeJson(jsonDoc, jsonData);
  // Extract values from the JSON object

  curtainData.storedHourOpen = jsonDoc["storedHourOpen"];
  curtainData.storedMinuteOpen = jsonDoc["storedMinuteOpen"];
  curtainData.storedHourClose = jsonDoc["storedHourClose"];
  curtainData.storedMinuteClose = jsonDoc["storedMinuteClose"];
  curtainData.repeatEveryDay = jsonDoc["repeatEveryDay"];
  curtainData.tempToggleValueCurtain = jsonDoc["tempToggleValueCurtain"];
  curtainData.sunToggleValueCurtain = jsonDoc["sunToggleValueCurtain"];
  curtainData.timeSet = jsonDoc["timeSet"];
  curtainData.storedDay = jsonDoc["storedDay"];
  curtainData.storedMonth = jsonDoc["storedMonth"];
  curtainData.storedYear = jsonDoc["storedYear"];

}
```

5.4.3.2 Open Close

➔ Following functions are get their datas directly from webpage. Not from controller.

➔ On the web page when the related button pressed, receiver will notice it. We use Ajax for this operaiton.

```
void openCurtain() {
  motorDirection = "left";
  startMotorLeft();
}
void closeCurtain() {
  motorDirection = "right";
  startMotorRight();
}
```

5.4.3.3 Read Sensors

➔ Curtain has 2 temperature sensor. One of them inside and the other one placed outside. It also has 1 light sensor which is also placed outside. We use CD4051 analog multiplexer to get all analog inputs from sensors since ESP only has 1 analog input.

➔ Following “updateSensorValues” functions reads analog inputs then convert it to necessary variable(celcius, voltage of ldr).

```
// sensor values connected to 4021
void updateSensorValues() {
  for (int channel = 0; channel < 3; channel++) {
    // Set the control pins to select the appropriate sensor channel
    selectSensorChannel(channel);
    if (channel == 0) {
      // Read the analog input value from the inside temp
      int multiplexerValue = analogRead(analogInputPin);
      curtainInTemp = multiplexerValue * 330 / 1024 - 273;
      delay(50);
    } else if (channel == 1) {
      // Read the analog input value from the outside temp
      int multiplexerValue = analogRead(analogInputPin);
      curtainOutTemp = multiplexerValue * 330 / 1024 - 273;
      delay(50);
    } else if (channel == 2) {
      int sensorValue = analogRead(analogInputPin);   // read the input of ldr
      curtainLdr = sensorValue * (3.3 / 1023.0);
    } else {
      //Serial.println("");
    }
  }
}
```

➔ Above function calls another function which is “selectSensorChannel”. It helps us to read the correct channel since CD4051 has 8 analog input pins.

```
void selectSensorChannel(int channel) {
  // Convert channel number to binary representation
  int pinAState = bitRead(channel, 0);
  int pinBState = bitRead(channel, 1);
  int pinCState = bitRead(channel, 2);

  // Set the control pins to select the specified channel
  digitalWrite(controlPinA, pinAState);
  digitalWrite(controlPinB, pinBState);
  digitalWrite(controlPinC, pinCState);
}
```

5.4.3.4 Start Motors

➔ According to the direction of the motor, we start the motor to open and close the curtain. Considering the effect of gravity, we make the opening speed one click faster than the closing speed.

```
void startMotorLeft() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 90);
}
void startMotorRight() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 50);
}
```

5.4.3.5 Limit Switch

➔ After we add “ezbutton” library and define which pin we have connected the switch we simply checkt it. when it pressed, it will call “stopMotor” function. We also store a data in the eeprom. Which is status of the curtain if its open it will store 1 and if its close 0 will be stored.

```
limitSwitchDown.loop();
limitSwitchUp.loop();
if (limitSwitchUp.isPressed() && motorDirection == "left") {
  stopMotor();
  EEPROM.put(0, 1);
}
if (limitSwitchDown.isPressed() && motorDirection == "right") {
  stopMotor();
  EEPROM.put(0, 0);
}
```

5.4.3.6 Main Loop

➔ In the main loop, first we have limit switch functions. We are calling them first because if I put some other lines of code before these functions, we may face with some delay problems.

➔ After that we call hanldeClient() function. We must call it inside the main loop to process and respond to the client's request.

➔ Then we have 4 functions. Functions are getting epoch time from internet, getting curtain datas from controller, sending temperature / sensor values to the controller and updating sensor values. These for functions are placed in some if statements. So they will not work continuously.

➔ Lastly, we have the curtains working logic. Below I explain the working logic of the main loop step by step:

- First, the value of "curtainData.timeSet" is checked. If this value is true, it means that the time has been adjusted.

- Next, the value of "curtainData.repeatEveryDay" is checked or whether the current date is the same as the year, month, and day values stored in curtainData. This determines whether the curtain movement is repeated every day.

- If the hours and minutes are equal to the opening hours and minutes stored in curtainData and the motor direction is not left, the motor direction is set to left and the motor starts to move to the left.

- If the hours and minutes are equal to the shutdown hours and minutes stored in the curtainData and the motor direction is not "right", the motor direction is set to "right" and the motor starts to move to the right.
- Next, if its summer and if the curtainData.tempToggleValueCurtain value is true, it continues. If its not it will scape the part that related with the temperature.

- After eneterd, If curtainOutTemp + 3.0 is less than curtainInTemp and the motor direction is not left, the motor direction is set to left and the motor starts to move to the left.

- If the curtainOutTemp value is greater than curtainInTemp + 3.0 and the motor direction is not "right", the motor direction is set to "right" and the motor starts to move to the right.

- If its not summer and curtainData.tempToggleValueCurtain is true, it continues.

- If curtainOutTemp + 3.0 is greater than curtainInTemp and the motor direction is not left, the motor direction is set to left and the motor starts to move to the left.

- If the curtainOutTemp value is less than curtainInTemp + 3.0 and the motor direction is not "right", the motor direction is set to "right" and the motor starts to move to the right.

- Next, the curtainData.sunToggleValueCurtain value is checked. If this value is true, the sun sensor is used. So the following lines will run.

- If the curtainLdr value is less than 0.10 and the motor direction is not "right", the motor direction is set to "right" and the motor starts to move to the right.

- If the curtainLdr value is greater than 0.40 and the motor direction is not left, the motor direction is set to left and the motor starts to move to the left.

- Finally, it occurs if the curtainData.timeSet, curtainData.sunToggleValueCurtain, and curtainData.tempToggleValueCurtain values are false and only the temperature will be set.

- If summer, the conditions related with temperature mentioned before are checked and the motor action takes place accordingly.

- If the not summer, the conditions related with temperature mentioned before are checked and the motor action takes place accordingly.

6 Conclusion

The development of the Smart Curtain system has successfully provided an innovative solution that allows users to control their curtains remotely. The system, which includes some sensors and advanced algorithms. It responds to external stimuli such as sunlight and temperature, providing an automatic and intelligent operation. The web-based interface can be used from any device connected to the home internet, providing seamless control.

Extensive testing and evaluations were conducted throughout the project to ensure the reliability of the system and its consistent performance in various environments. The integration of sensors and algorithms not only improves the functionality of the system, but also demonstrates its capacity to intelligently adapt to changing conditions (for example, it performs different actions depending on the value it reads from the temperature sensor in summer and winter). This smart and efficient approach distinguishes the Smart Curtain system as a superior solution.

The user-friendly web-based interface is simply designed with accessibility in mind, giving users effortless access by entering their IP address. Users accessing the Web Page have control over their curtains as they wish. Users get a personalized and convenient user experience by allowing them to tailor the system to their specific needs. The integration of the Smart Curtain system with the web interface provides an adaptable and versatile solution that can integrate seamlessly into various settings.

In summary, the Smart Curtain system successfully fulfills its objectives of providing a reliable and convenient method to control curtains. With its advanced sensor integration, smart algorithms and user-friendly web-based interface, it stands out as an efficient and smart solution suitable for a wide range of applications. The achievements of the project highlight the potential of the Smart Curtain system to transform the way curtains are managed and contribute to enhanced user comfort.

6.4 Benefits

The development of a smart curtain system that can be controlled through a web interface and integrated with sensors has the potential to provide a range of benefits for both the user and the developer.

a. Benefits to users :

1. User will be able to open and close their curtain with the touch of a button.

2. User can set a specific time for opening and closing their curtains.

3. Improved comfort: The smart curtain allows a comfortable living space throughout the day.

4. User can monitor the temperature of room on the screen or on the webpage.

b. Benefits to me :

1. The project presents an opportunity to gain valuable skills in web development.

2. Skill development: Creating this project enhances my knowledge and skills in embedded systems, and programming.

3. Practical application: The project demonstrated my ability to design and implement a functional smart home automation system, and I gained practice in this field.

4. Gaining skills on network communication.

Overall, the successful development and implementation of a smart curtain system has the potential to improve the quality of life for both the user and the developer.

6.5 Ethics

There has been a discrepancy between the rapid adoption of smart home devices such as smart curtain and the advancement of ethical research and the average person's understanding of privacy. Ensuring privacy in the home involves both physical and informational protection. Therefore, while there are not so many issues of data protection and or discrimination, below are some of the ethics of the project:

- Respect for user privacy: The smart curtain system should be designed to protect the privacy of users, including the use of secure connections and the handling of any personal data collected by the system.
- Informed consent: Users should be fully informed about the capabilities and limitations of the smart curtain system before they agree to use it. This may include providing detailed information about how the system works, any potential risks or limitations, and how personal data is collected and used.
- Data protection: The smart curtain system should be designed to protect the confidentiality, integrity, and availability of any personal data collected by the system. This may include the use of secure servers and storage systems, as well as regular backups and updates to protect against cyber threats.
- Transparency: The smart curtain system should be transparent about its capabilities, limitations, and any potential risks or concerns associated with its use. This may include providing regular updates and notifications to users about the status of the system and any necessary maintenance or repairs.
- User control: Users should have control over how the smart curtain system is used and any personal data that is collected by the system. This may include the ability to opt out of certain features or to delete their account and any associated data.

Responsible use: Users should be encouraged to use the smart curtain system responsibly and to respect the rights and privacy of others. This may include not using the system to engage in activities that are illegal or harmful to others.

Why did I choose this project?

The biggest reason I chose this project is that it is a solution to a problem I have experienced myself.

I've always had a hard time waking up in the morning in a dark environment. When I go to bed without closing the curtain and want to wake up in the morning with daylight, the light from the street lamps illuminates and disturbs the interior of the room before I go to sleep.

I was continuing to have the same problem while staying in the hotel rooms in a hotel where I worked in the USA. When I realized that the curtain of the rooms on the luxury floor of the hotel were motorized, I thought of such a project. The curtain in the hotel room could only be opened and closed with the button on the wall. I wanted to turn it into a smart curtain that is sensitive to daylight and can be controlled remotely. When I researched this project that came to my mind, I realized that there was no such project on the internet and I decided to do it.

6.6 Future Works

Yes, I intend to continue the project. I plan to make the project more robust and reliable by making some additions in the upcoming summer period. Afterwards, if I can't find a business opportunity that I want and I have enough budget, I can market the project by making it on a real curtain.

Necessary measures can be taken to prevent data loss on the Internet. In order to prevent data loss that occurs during the hours when the internet is used heavily, we can perform tests on a stable internet, check if the data is not sent, and ensure that it is sent again. We can define deadlines for sending the submitted data.

We can add button on controller. When we click the button, we can switch between the curtains and the information of whether the curtains are open and the outside temperature of the relevant curtain can be printed on the screen. At the same time, while one of the curtains is opening or closing, the name of the relevant curtain and opening/closing information can be printed. We can even add an on and off animation.

I couldn't do it because my current modem doesn't support it, but with using a different modem, we can define fixed IP addresses to ESPs, so IP addresses will never change and we will not have to deal with the setup part all the time.

If any of the curtain have a power outage or internet problem. If there is no data exchange between the controller and the curtian for a certain period of time, the information that the relevant curtain is inactive can be printed on the oled screen.

The feature of opening the curtains halfway and adjusting the striped curtains to receive sunlight when closed.

By measuring the distance that the motor should run, depending on what is desired, the gap between the striped curtains can be opened by opening the curtain halfway or opening and closing it a little.

After becoming completely stable and reliable with the web, mobile applications such as ios and android can be developed.

The cables on the breadboard can be arranged and put into the box.

7 References

[1] : Serena Smart Shades | Lutron. (n.d.). Retrieved from https://www.lutron.com/en-US/Products/Pages/ShadingSystems/SerenaShades/Overview.aspx

[2] : myLink for Smartphones and Tablets | Somfy. (n.d.). Retrieved from https://www.somfysystems.com/en-us/products/1811403/mylink-rts-smartphone-and-tablet-interface-120v-ac

[3] : Amazon.com. (n.d.). Retrieved from https://www.amazon.com/Upgraded-Version-SwitchBot-Curtain-Electric/dp/B09Y5ZF81H/ref=sr_1_4?keywords=smart+curtains&qid=1671325855&sr=8-4&th=1CurtainElectric/dp/B09Y5ZF81H/ref=sr_1_4?keywords=smart%2Bcurtains&qid=1671325855&sr=8-4&th=1

[4] : SONTE - About us. (n.d.). Retrieved from https://sonte.com/about.html

[5] : Easy to install motors for shades & blinds. (n.d.). Retrieved from https://www.somasmarthome.com/

[6] : The best smart blinds and shades of 2023. (n.d.). Retrieved from https://www.zdnet.com/home-and-office/smart-home/best-smart-blinds/

[7] : (n.d.). Retrieved from https://www.linkedin.com/pulse/ethics-smart-home-technology-malik-williams/

[8] : Motorized Shades - Affordable Smart & Electric Shades. (n.d.). Retrieved from https://www.ikea.com/us/en/cat/electric-blinds-44531/

[9] : IEEE. (n.d.). ETHICALLY ALIGNED DESIGN. The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, 266. https://standards.ieee.org/wpcontent/uploads/import/documents/other/ead_v2.pd

[10] : Stip, E. (2005, April). Environmental Cognitive Remediation in Schizophrenia: Ethical Implications of "Smart Home" Technology. Original Research, 11. https://journals.sagepub.com/doi/pdf/10.1177/070674370505000509

[11] : Instructables. (2017, August 7). Interface L298N Using NodeMCU. Retrieved from https://www.instructables.com/Interface-L298N-Using-NodeMCU/

[12] : Instructables. (2017, July 13). NodeMCU With LDR. Retrieved from https://www.instructables.com/NodeMCU-With-LDR/

[13] : The LM335 and it's serieresistor. (n.d.). Retrieved from https://arduinodiy.wordpress.com/2017/04/17/the-lm335-and-its-serieresistor/

[14] : (n.d.). Retrieved from https://assets.nexperia.com/documents/data-sheet/HEF4051B.pdf

[15] : AJAX Introduction. (n.d.). Retrieved from https://www.w3schools.com/js/js_ajax_intro.asp

[16] : Alerts. (n.d.). Retrieved from https://getbootstrap.com/docs/5.0/components/alerts/

[17] : JavaScript HTML DOM. (n.d.). Retrieved from https://www.w3schools.com/js/js_htmldom.asp

[18] : DIYables Limit Switch for Arduino, ESP32, ESP8266, Raspberry Pi. (n.d.). Retrieved from https://diyables.io/products/limit-switch

[19] : ESP8266 0.96 inch OLED Display with Arduino IDE | Random Nerd Tutorials. (n.d.). Retrieved from https://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/

[20] : ESP8266 Client-Server Wi-Fi Communication Between Two Boards (NodeMCU) | Random Nerd Tutorials. (n.d.). Retrieved from https://randomnerdtutorials.com/esp8266-nodemcu-client-server-wi-fi/

[21] : Filesystem — ESP8266 Arduino Core 3.1.2-11-g57fa6cdc documentation. (n.d.). Retrieved from https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html

[22] : CH340 Drivers for Windows, Mac and Linux. (n.d.). Retrieved from https://sparks.gogo.co.nz/ch340.html

[23] : ESP8266 Pinout Reference: Which GPIO pins should you use? | Random Nerd Tutorials. (n.d.). Retrieved from https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/

[24] : Getting Started with ESP8266 NodeMCU Development Board| Random Nerd Tutorials. (n.d.). Retrieved from https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/

[25] : Build an ESP8266 Web Server - Code and Schematics (NodeMCU) | Random Nerd Tutorials. (n.d.). Retrieved from https://randomnerdtutorials.com/esp8266-web-server/

[26] : Install ESP8266 Filesystem Uploader in Arduino IDE | Random Nerd Tutorials. (n.d.). Retrieved from https://randomnerdtutorials.com/install-esp8266-filesystem-uploader-arduino-ide/

[27] : Instructables. (2017, June 26). Interface LM35 With NodeMCU. Retrieved from https://www.instructables.com/Interface-LM35-With-NodeMCU/

[28] : Instructables. (2023, April 6). PocketServer: Turn Your ESP8266 Into an HTML Web Server (and Connect It to the Internet). Retrieved from https://www.instructables.com/PocketServer-Turn-Your-ESP8266-Into-an-HTML-Web-Se/